COMMAND PATTERN SUMMARY – TEAM FAITHFUL

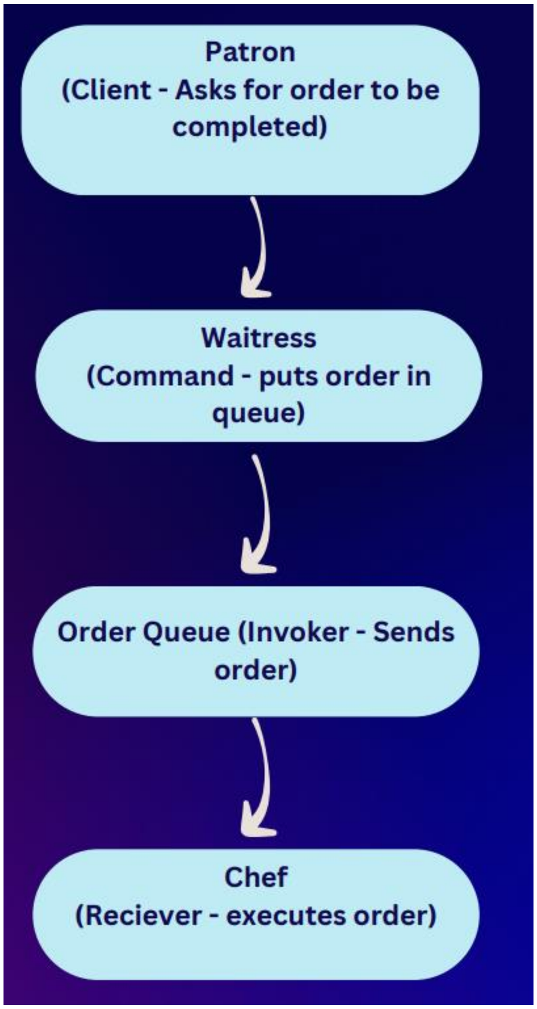
The command pattern is a data-driven design pattern and falls under the behavioral pattern category. A request is wrapped under an object as a command and passed to the invoker object. The invoker object looks for the appropriate object that can handle this command and passes the command to the corresponding object which executes the command.

The Command design pattern encapsulates a request as an object, allowing users to parameterize clients with queues, requests, and operations. It allows for the separation of concerns between an object that invokes an operation, called the client, and the object that operates, called the receiver.

History:

The Command Pattern has been around for a long time and is deeply rooted in object-oriented design principles. It became widely known after being described in the famous book "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (commonly known as the Gang of Four, or GoF)

Real World Example:



Patron (Client - Asks for order to be completed)

Waitress (Command - puts order in queue)

Order Queue (Invoker - Sends order)

Chef (Reciever - executes order)

Objects in Command:

- Command
    - Sends command to receiver
- Receiver
    - Performs command task
- Invoker
    - Invokes command, bookeeps execution
- Client
    - Contains command and invoker objects, sends command object to invoker

Advantages:

The main advantage of the command pattern is that the invoker and receiver are low coupled objects. This means that any changes to one will not directly affect the other. Another advantage of using design patterns in general is good organization and readability in code.

Disadvantages:

Disadvantage of Command The main drawback is that Command requires more objects to invoke a command than just a single method object.

Why & How to Implement:

If you find your code using a lot of if-else statements to branch into different functions, consider using the Command Pattern. Keys to implementing: Separate your code into an invoker, command objects, and receiver so that the code is low coupled and more readable under a client function.

Sample Code